BITP 3123 Distributed Application Development

Lab Week 10

Schedular

By

Emaliana Kasmuri

FTMK, UTeM

Learning Outcome

At the end of this lab exercise, the student should be able to:-

1. Execute delayed task program.
2. Implement delayed task program using classes and interfaces from `java.util.concurrent`.
3. Execute a scheduled task from a Spring Boot application.
4. Implement a scheduled task for a Spring Boot application.

Exercise 1: Setting Up Demo Environment

1. Download **scheduledtask1.zip** from ulearn.
2. Unzip **scheduledtask1.zip**. It should produce a folder name **scheduledtask1**.
3. Create a new Java project named **schedulethread**.
4. Import **scheduledtask1** as a package in the project.
5. Expand **scheduledtask1** from the Project Explorer. The content of the package should be similar as shown in Figure 1.

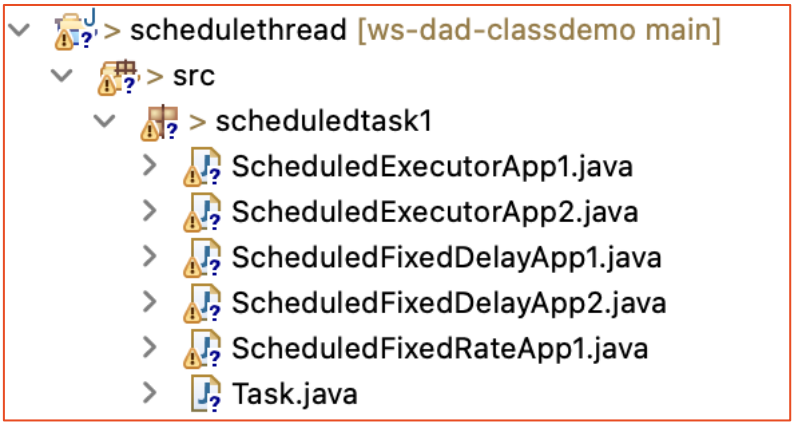

Figure 1: A list of classes in package scheduledtask1

Exercise 2: Executing a Delayed Task using ScheduledExecutorService

1. Double click class `ScheduledExecutorApp1.java` from package `scheduledtask1`.
2. Observe the program line-by-line.
3. Make a cross reference to `Task.java` at the line where a `Runnable` object is created.
4. Identify the method used from this link in `ScheduledExecutorApp1.java`.
5. Map the parameter used in the method to the value specified in `ScheduledExecutorApp1.java`.
6. Run `ScheduledExecutorApp1.java`.
7. Observe the output.
8. Compare the execution time.

Exercise 3: Changing the Time of Delayed Task

1. Double click class `ScheduledExecutorApp1.java` from package `scheduledtask`.
2. Change the value of delayed time for task1 to 5.
3. Save the class.
4. Run `ScheduledExecutorApp1.java`.
5. Observe the output.
6. Compare the execution time.
7. `ScheduledExecutorApp1.java` for several time to observe the output.
8. Change the value of delayed time for task1 to 15.
9. Repeat step 3 until 7.

Exercise 4: Implement a Delayed Task using ScheduledExecutorService

There two activities in this exercise.

Activity 4.1: Define a Task Class

1. Create a task class that will compute a summation of 100 random integer numbers.
2. The class should have a private attribute that represents the task name.
3. The constructor will initialize the task name according to the parameter value.
4. The task class should display the result of the computation.
5. The task class should display the execution before the computation starts.
6. Name the task class appropriately.

Activity 4.2: Create a Delayed Task using ScheduledExecutorService

1. Create a Java class with a `main( )` method. Name the class appropriately.
2. The class will execute the task defined in 3.1 as a delayed task using `ScheduledExecutorService`. Refer to ScheduledExecutorApp1.java.
3. The task should begin 5 seconds after the main( ) program starts.
4. The class should record the execution time.
5. Save the class.
6. Fix any errors.
7. Execute the class.

Exercise 5: Executing Multiple Delayed Tasks using ScheduledExecutorService

1. Double click class ScheduledExecutorApp2.java from package scheduledtask1.
2. Observe the program line-by-line.
3. Make a cross reference to Task.java at the line where multiple Runnable objects are created.
4. Run ScheduledExecutorApp2.java.
5. Observe the output.
6. Compare the execution time.
7. Execute the Run ScheduledExecutorApp2.java for several time to observe the output.

Exercise 6: Changing Delayed Time of Multiple Delayed Tasks

10. Double click class ScheduledExecutorApp2.java from package scheduledtask.
11. Change the value of delayed time for task1, task2 and task3 to 5.
12. Save the class.
13. Run ScheduledExecutorApp2.java.
14. Observe the output.
15. Compare the execution time.
16. Execute the Run ScheduledExecutorApp2.java for several time to observe the output.

Exercise 7: Implement Multiple Delayed Task

Activity 7.1: Define a Task Class

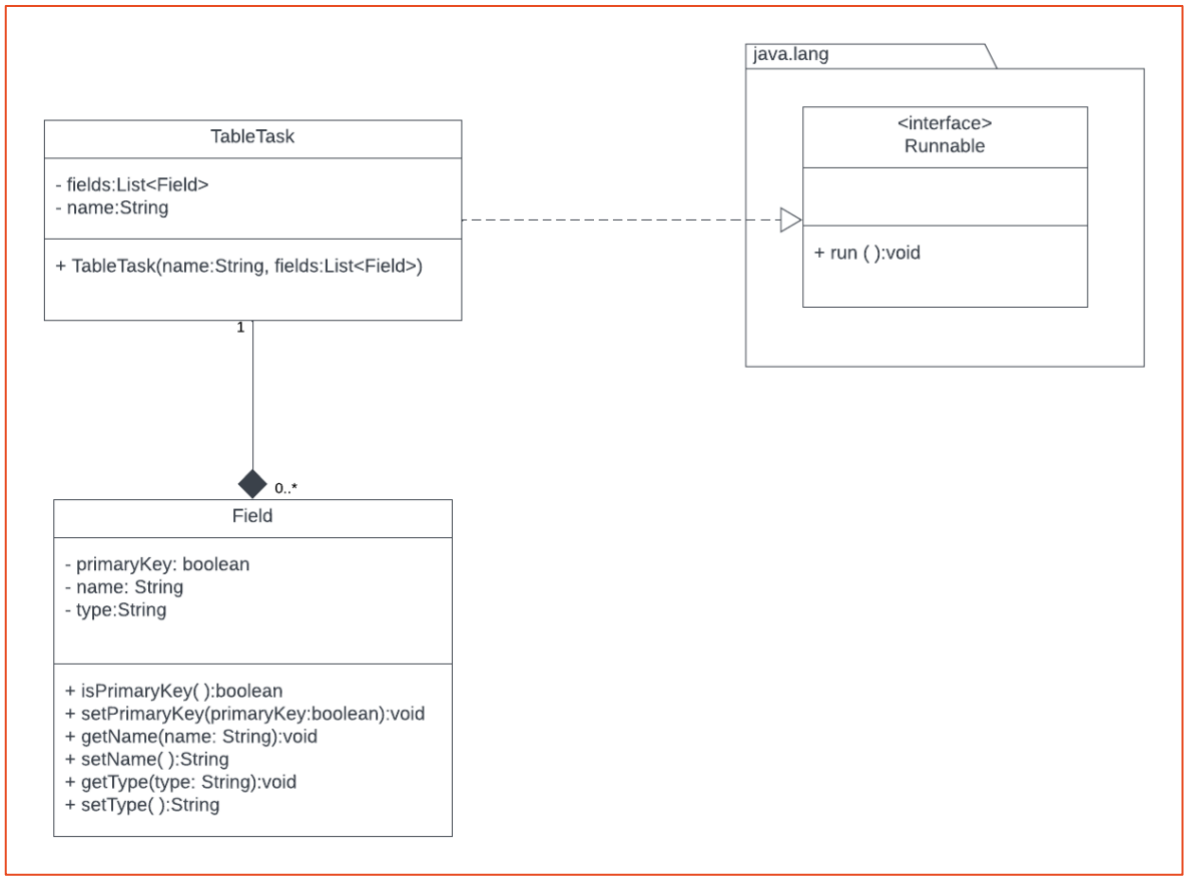

Figure 2: Relations of task class

Figure 2 shows a relation TableTask with an entity class and an interface. TableTask is a task class that represents generation of SQL statements for a table data definition language (DDL) and data manipulation language (DML). The DDL and DML will generate the SQL statement using the table fields. Attribute fields represent the fields of a table in the database, while the other attribute represents the name of the table.

1. Define the task class shown in Figure 2.
2. Add the methods to generate SQL statement the following specification.
    a. To create a table and its field.
    b. To add a new record into the table.
    c. To update an existing record from the using the primary key

d. To select all records in the table.
e. To select a specific record in the table using the primary key value.
f. To delete a specific record from the table using the primary key value.
g. To drop the table.

3. The task should implement the following specification:
    a. Display the SQL statements to create and drop the table.
    b. Display the SQL statements to add, update, select and delete records from the table.
    c. Display the execution time.

Activity 7.2: Creating Multiple Delayed Task

1. Create a Java class with a main( ) method to generate SQL statements for four different tables.
2. The class should schedule it as four delayed tasks at different delayed time.

Exercise 8: Executing a Scheduled Job using CRON

1. Download scheduleddemo.zip.
2. Unzip scheduleddemo.zip.
3. Import as a new Maven project in Eclipse.
4. Expand package my.edu.utem.ftmk.dad.scheduledcrondemo from the project.
5. Run ScheduledcrondemoApplication.java.
6. Observe the output.

Keep all output and codes intact. More exercises will be coming.